



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/050,387	01/16/2002	Nicolai Kosche	004-7051	6183
22120	7590	10/21/2004	EXAMINER	
ZAGORIN O'BRIEN & GRAHAM, L.L.P. 7600B N. CAPITAL OF TEXAS HWY. SUITE 350 AUSTIN, TX 78731			VO, TED T	
			ART UNIT	PAPER NUMBER
			2122	

DATE MAILED: 10/21/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	<b>Application No.</b>	<b>Applicant(s)</b>
	10/050,387	KOSCHE ET AL.
	<b>Examiner</b>	<b>Art Unit</b>
	Ted T. Vo	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
**Period for Reply**

**A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM  
 THE MAILING DATE OF THIS COMMUNICATION.**

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) Responsive to communication(s) filed on 16 January 2002.
- 2a) This action is FINAL.                    2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) Claim(s) 1-69 is/are pending in the application.
- 4a) Of the above claim(s) 54-60 is/are withdrawn from consideration.
- 5) Claim(s) \_\_\_\_\_ is/are allowed.
- 6) Claim(s) 1-53 and 61-69 is/are rejected.
- 7) Claim(s) \_\_\_\_\_ is/are objected to.
- 8) Claim(s) 54-60 are subject to restriction and/or election requirement.

**Application Papers**

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on \_\_\_\_\_ is/are: a) accepted or b) objected to by the Examiner.  
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
    - a) All    b) Some \* c) None of:
      1. Certified copies of the priority documents have been received.
      2. Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
      3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) Notice of References Cited (PTO-892)
- 2) Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
 Paper No(s)/Mail Date 2/10/03, 4/21/03
- 4) Interview Summary (PTO-413)  
 Paper No(s)/Mail Date 10/14/04
- 5) Notice of Informal Patent Application (PTO-152)
- 6) Other: \_\_\_\_\_

Art Unit: 2122

#### **DETAILED ACTION**

1. This action is in response to the application filed on 01/16/02.  
Claims 1-69 are pending in the application.

#### ***Election/Restrictions***

2. Restriction to one of the following inventions is required under 35 U.S.C. 121:
  - I. Claims 1-33, 34-53, 61-69, drawn to method, optimizing compiler and computer program product, classified in class 717, subclass 148.
  - II. Claims 54-60, drawn to a method, classified in class 717, subclass 159.

The inventions are distinct, each from the other because of the following reasons:

Inventions I and II are unrelated. Inventions are unrelated if it can be shown that they are not disclosed as capable of use together and they have different modes of operation, different functions, or different effects (MPEP § 806.04, MPEP § 808.01).

In the instant case, Claims of Group I provide code preparing in an execution process and optimize the code, and Claims of Group II are in another stage of compiling as code inserting relating processor or pipelining.

Because these inventions are distinct for the reasons given above and have acquired a separate status in the art as shown by their different classification, restriction for examination purposes as indicated is proper.

Because these inventions are distinct for the reasons given above and the search required for Group I is not required for Group II, restriction for examination purposes as indicated is proper.

During a telephone conversation with Mr. O'Brien, Attorney Reg. No. 40107, on 10/13/04, a provisional election was made with traverse to prosecute the invention of Group I, claims 1-53 and 61-69. Affirmation of this election must be made by Applicants in replying to this Office action. Claims 54-60 are

withdrawn from further consideration by the examiner, 37 CFR 1.142(b), as being drawn to a non-elected invention.

***Specification***

3. This specification contains URL code (text block 007). The disclosure is objected to because it contains an embedded hyperlink and/or other form of browser-executable code. Applicants are required to delete the embedded hyperlink and/or other form of browser-executable code. See MPEP § 608.01  
Uncompleted information in text blocks 0001, 0026, etc., which identifies to unknown US patent applications requires updating when the information is available. If the information is an US patent, the US Patent Number is required.

***Claim Rejections - 35 USC § 112***

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:  
The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.
5. Claim 33 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The functionality of Claim 33 is unclear because it is dependent on itself. This type of dependency is indefinite.

***Claim Rejections - 35 USC § 102***

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 1-53, 61-69 are rejected under 35 U.S.C. 102(b) as being anticipated by Armstrong, "HotSpot: A new breed of virtual machine", Java World, <http://www.javaworld.com>, 1998.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Armstrong discloses,

*A code preparation method comprising:*

*identifying at least one operation in first executable instance of code* (Page 4, second paragraph, "When bytecodes are first loaded,...": referring "When a method is found");  
*executing the first executable instance and responsive to detection of an execution event* (page 4, Second paragraph, referring "runtime"), *associating a corresponding execution characteristic* (page 4, Second paragraph, referring "profiler") *with a corresponding identified one of the operations* (page 4, Second paragraph, referring "method"/"HotSpot"); and

*preparing a second executable instance of the code based, at least in part, on the association between the execution characteristic and the identified operation* (page 4, second paragraph, referring "Every future call to that method uses the native machine instructions produced by the compiler":  
*preparing a second executable instance*),

See pages 3 and 4, description of How the dynamic compiler works.

As per Claim 2: Armstrong discloses,

*The method of claim 1, wherein the operation identification is consistent between the first executable instance and the preparation of the second executable instance.*

See page 4, second paragraph, referring "Every future call (*is consistent between*) to that method (*first executable instance*) uses the native machine instructions (*the second executable instance*) produced by the compiler"

As per Claim 3: Armstrong discloses,

*The method of claim 2, wherein the consistency of operation identification is maintained from preparation of the first executable instance to preparation of the second executable instance.*

See page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler" and the whole description of description of How the dynamic compiler works.

As per Claim 4: Armstrong discloses,

*The method of claim 1, wherein same unique identification numbers are assigned to corresponding operations of the first executable and the second executable.*

See page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler" and the whole description of description of How the dynamic compiler works, where *same unique identification* has been read from the correspondence between a method that has hotspot and is executed in the first time and the native optimized code (prepared code) executed in the place of that method.

As per Claim 5: Armstrong discloses,

*The method of claim 4, wherein the execution characteristic is associated with the unique identification number.*

see description of How the dynamic compiler works, the correspondence between a method that has hotspot and is executed in the first time and the native optimized code (prepared code) executed in the place of that method.

As per Claim 6: Armstrong discloses,

*The method of claim 4, wherein the unique identification numbers and their assignment to operations are maintained throughout any optimizations or code transformations performed in preparation of the first executable.*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", and the whole description of description of How the dynamic compiler works.

As per Claim 7: Armstrong discloses,

*The method of claim 6, wherein the maintenance of the unique identification number assignments include further assigning the unique identification number to a copy when an operation is copied as part of a code transformation or optimization.*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", and the whole description of description of How the dynamic compiler works.

As per Claim 8: Armstrong discloses,

*The method of claim 6, wherein the maintenance of the unique identification number assignments includes removing an assignment when the assigned operation is removed as part of a code transformation or optimization*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler". Thus, the Hotspot compiler, the optimization, has removed the assignment that calls the method, and assigned such a call to the optimized native code in its future execution.

As per Claim 9: Armstrong discloses,

*The method of claim 1, wherein the associating of the corresponding execution characteristic includes encoding aggregated hardware event information in an extended definition of an instruction instance for use in the preparation of the second executable instance.*

as providing profiling in the HotSpot compiler of the figure in page 3, where profiling of Hotspot is known for using counters (hardware event information) to detect a low performance, threshold, caused by a hotspot. See pages 3-4, second paragraph, "the whole description of description of How the dynamic compiler works, and Profiling and compiling heuristics.

As per Claim 10: Armstrong discloses,

*the method of claim 1, wherein the identified operation is a memory access instruction*

in the HotSpot compiler mechanism: referring native instructions.

As per Claim 11: Armstrong discloses, based on the broad limitation,

*The method of claim 1, wherein the execution characteristic includes a cache miss likelihood.*

in the HotSpot compiler mechanism because *cache miss* also causes hotspot.

As per Claim 12: Armstrong discloses,

*The method of claim 1, wherein the preparation includes inserting one or more prefetch operations in the code prior to the identified operation to exploit latency provided by servicing of a cache miss by the identified operation.*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", where "uses the native machine instructions" is an act of inserting and prefetching, and where the method that is replaced by native machine instructions has hotspot such as looping, cache miss etc.

As per Claim 13: Armstrong discloses,

*The method of claim 1, further comprising: preparing the first executable instance.*

see page 4, second paragraph, referring the profiling within the method at the first time before the future execution of native machine instructions.

As per Claim 14: Armstrong discloses,

*The method of claim 13, wherein the preparation of the first executable instance includes substantially all optimizations operative in the preparation of the second executable.*

see page 4, first and second paragraphs, referring the act in optimizing hotspots to produce the native instructions (second executable).

As per Claim 15: Armstrong discloses,

*The method of claim 14, wherein execution of the first executable instance corresponds substantially with execution of an executable instance of code prepared without the identifying.*

see page 4, first and second paragraphs.

As per Claim 16: Armstrong discloses,

*The method of claim 14, whereby execution of the first executable instance sufficiently corresponds to that in an expected execution environment, so that the execution characteristic is applicable to the preparation of the second executable.*

see pages 3-4, whole description of How dynamic compiler works.

As per Claim 17: Armstrong discloses,

*The method of claim 13, wherein the preparation of the first executable instance forgoes certain optimizations performed, after use of the association between the execution characteristic and the identified instruction, by the further preparing.*

see pages 3-4, referring profiling, and whole description of How dynamic compiler works.

As per Claim 18: Armstrong discloses,

*The method of claim 13, wherein the preparation of the first executable instance includes compilation of the code.*

see page 4, referring Profiling and compiling heuristics.

As per Claim 19: Armstrong discloses,

*The method of claim 1, wherein both the first and the second executable instances are compiled instances of the code.*

see pages 3-4, whole description of How dynamic compiler works.

As per Claim 20: Armstrong discloses,

*The method of claim 1, wherein the second executable instance is an optimization of the first executable instance.*

see page 4, second paragraph.

As per Claim 21: Armstrong discloses,

*The method of claim 1, wherein the preparing includes optimizations forgone in the first executable instance.*

see page 4, Profiling and compiling heuristics.

As per Claim 22: Armstrong discloses,

*The method of claim 1, wherein the preparation of the second executable instance includes optimizations forgone in preparation of the first executable instance.*

see pages 3-4, whole description of How dynamic compiler works; see page 4, Profiling and compiling heuristics.

As per Claim 23: Armstrong discloses,

*The method of claim 1, wherein at least the preparing is performed by an optimizing compiler.*

see the figure in page 3.

As per Claim 24: Armstrong discloses,

*The method of claim 1, wherein at least the preparing is performed by a binary translator.*

see the figure in page 3.

As per Claim 25: Armstrong discloses,

*The method of claim 1, wherein at least the preparing is performed by a binary rewriter.*

see the figure in page 3.

As per Claim 26: Armstrong discloses,

*The method of claim 1, wherein at least the preparing is performed by a binary optimizer.*

see the figure in page 3.

As per Claim 27: Armstrong discloses,

*The method of claim 1, wherein at least the preparing is performed by a just-in-time (JIT) compiler.*

see the figure in page 3.

As per Claim 28: Armstrong discloses,

*The method of claim 1, wherein the associating of the corresponding execution characteristic includes aggregating contributions of plural instances of the execution event.*

see the figure in page 3.

As per Claim 29: Armstrong discloses,

*The method of claim 1, wherein the associating of the corresponding execution characteristic includes backtracking from a point in the code that coincides with delayed detection of the execution event.*

see the figure in page 3.

As per Claim 30: Armstrong discloses,

*The method of claim 1, wherein the associating of the corresponding identified one of the operations includes reading or receiving a computer readable encoding of an event profile.*

see the figure in page 3 and see page 4: Profiling and compiling heuristics.

As per Claim 31: Armstrong discloses,

*The method of claim 1, wherein the associating of the corresponding execution characteristic includes reading or receiving a computer readable encoding of an event profile.*

see the figure in page 3 and see page 4: Profiling and compiling heuristics.

As per Claim 32: Armstrong discloses,

*The method of claim 1, further comprising: preparing the second executable instance as a computer program product for distribution, transmission or execution.*

see the figure in page 3 and the computer program product is referred to Java program.

As per Claim 33: Armstrong discloses,

*The method of claim 33, wherein the computer program product is encoded in one or more computer readable media selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.*

see the figure in page 3 and the computer program product is referred to Java program.

As per Claim 34: Claim 34 recites a compiler that performs the steps corresponding to the limitation of Claim 1. See the rationale on Claim 1 above.

As per Claim 35: Claim 35 recites a compiler that performs the steps corresponding to the limitation of Claim 2. See the rationale on Claim 2 above.

As per Claim 36: Claim 36 recites a compiler that performs the steps corresponding to the limitation of Claim 6. See the rationale on Claim 6 above.

As per Claim 37: Claim 37 recites a compiler that performs the steps corresponding to the limitation of Claim 7. See the rationale on Claim 7 above.

As per Claim 38: Claim 38 recites a compiler that performs the steps corresponding to the limitation of Claim 8. See the rationale on Claim 8 above.

As per Claim 39: Claim 39 recites a compiler that performs the steps corresponding to the limitation of Claim 12. See the rationale on Claim 12 above.

As per Claim 40: Regarding limitation,

*"The optimizing compiler of claim 34, wherein the transformations include insertion of one or more non-faulting loads",*

see page 4, second paragraph, "Every future call to that method uses the native machine instructions produced by the compiler", where "uses the native machine instructions" is an act of inserting, and see "native machine instructions" (*non-faulting loads*).

As per Claim 41: Claim 41 recites a compiler that performs the steps corresponding to the limitation of Claim 13. See the rationale on Claim 13 above.

As per Claim 42: Claim 42 recites a compiler that performs the steps corresponding to the limitation of Claim 14. See the rationale on Claim 14 above.

As per Claim 43: Claim 43 recites a compiler that performs the steps corresponding to the limitation of Claim 11. See the rationale on Claim 11 above.

As per Claim 44: Claim 44 recites a compiler that performs the steps corresponding to the limitation of Claim 10. See the rationale on Claim 10 above.

As per Claim 45: Claim 45 recites a compiler that performs the steps corresponding to the limitation of Claim 14. See the rationale on Claim 14 above.

As per Claim 46: Regarding,

*The optimizing compiler of claim 34, embodied as part of a binary translator*  
referring to the figure in page 3.

As per Claim 47: Regarding,

*The optimizing compiler of claim 34, embodied as part of a binary rewriter.*  
referring to the figure in page 3.

As per Claim 48: Regarding,

*The optimizing compiler of claim 34, embodied as part of a binary optimizer.*

referring to the figure in page 3.

As per Claim 49: Regarding,

*The optimizing compiler of claim 34, embodied as a just-in-time (JIT) compiler.*

referring to the figure in page 3.

As per Claim 50: Claim 50 recites a compiler that performs the steps corresponding to the limitation of Claim 6. See the rationale on Claim 6 above.

As per Claim 51: Regarding,

*The optimizing compiler of claim 34, wherein the optimizing compiler identifies one or more memory access instructions in the first executable instance of the computer program code; and wherein the run-time information encodes respective execution characteristics for respective ones of the identified memory access instructions.*

see pages 4-5, referring to Profiling and compiling heuristics.

As per Claim 52: Regarding,

*The optimizing compiler of claim 34, wherein collection of the run-time information includes aggregation of execution event information and association of the aggregated information with memory access instructions identified in the first executable instance of the computer program code.*

see pages 4-5, referring to Profiling and compiling heuristics.

As per Claim 53: Regarding,

*The optimizing compiler of claim 34, encoded in one or more computer readable media selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.*

see rationale in Claim 1.

As per Claim 61: Regarding,

*A computer program product encoded in one or more computer readable media, the computer program product comprising: a first execution sequence; and an information encoding associating an execution event with at least some operation of the first execution sequence, the associated execution event based at least in part on an execution profile of the first execution sequence of operations, wherein consistency of the association is maintained from preparation of the first executable instance for preparation of a second executable instance.*

see rationale in Claim 1.

As per Claim 62: Regarding,

*The computer program product of claim 61, wherein the execution event is a cache miss likelihood.*

see rationale in Claim 11.

As per Claim 63: Regarding,

*The computer program product of claim 61, wherein the associated operation is a memory access operation.*

see rationale in Claim 10.

As per Claim 64: Regarding,

*The computer program product of claim 61, employed in an data structure of an optimizing compiler in preparation of an optimized instance of the execution sequence of operations, wherein the optimized instance includes one or more prefetch operations placed before particular ones of the memory access operations for which the associated information encoding indicates a cache miss likelihood.*

see rationale in Claim 12.

As per Claim 65: Regarding,

*The computer program product of claim 61, wherein the one or more computer readable media are selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.*

see rationale in Claim 1.

As per Claim 66: Claim 66 recites an apparatus that performs the steps corresponding to the limitation of Claim 1. See the rationale on Claim 1 above.

As per Claim 67: Regarding,

*The apparatus of claim 66, wherein the identifying includes producing a table of tags and operation addresses.*

as part of profiling, in page 4, Profiling and compiling heuristics.

As per Claim 68: Regarding,

*The apparatus of claim 66, wherein information for the identifying is encoded in a file or communications channel read by the means for collecting.*

as part of profiling, in page 4, Profiling and compiling heuristics.

As per Claim 69: Regarding,

*The apparatus of claim 66, further comprising: means for preparing the first executable instance of the computer program code.*

as part of profiling, in pages 3-4, How the dynamic compiler work and Profiling and compiling heuristics.

### ***Conclusion***

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

**Kwong et al., US Pat. No. 6,289,506 B1**, discloses performance of Java Virtual Machine using precompiled code.

**Venners et al., "The Hotspot Virtual Machine"**, discloses improving Java performance of how Hotspot attempts to eliminate bottlenecks.

**Alpern et al., "The Jalapeno Virtual Machine"**, discloses a structure a Java Virtual Machine.

**White paper, "The Java HotSpot Engine Architecture"**,

[Http://java.sun.com/products/whitepaper.html](http://java.sun.com/products/whitepaper.html), discloses a Java virtual machine for HotSpot.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (703) 308-9049. The examiner can normally be reached on 8:00AM to 5:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

After October 28, 2004, examiner can be reached at new telephone number (571) 272-3706 and the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3694.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TED T. VO

TTV  
Patent Examiner  
Art Unit 2122  
October 14, 2004